44

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/676,552	09/30/2000	MICHAEL GINSBERG	MS1-0859US	6912

22801 7590 06/20/2007
LEE & HAYES PLLC
421 W RIVERSIDE AVENUE SUITE 500
SPOKANE, WA 99201

EXAMINER	
BULLOCK JR, LEWIS ALEXANDER	

ART UNIT	PAPER NUMBER
2195	

NOTIFICATION DATE	DELIVERY MODE
06/20/2007	ELECTRONIC

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Notice of the Office communication was sent electronically on above-indicated "Notification Date" to the following e-mail address(es):

lhptoms@leehayes.com

Office Action Summary	Application No.	Applicant(s)
	09/676,552	GINSBERG, MICHAEL
	Examiner	Art Unit
	Lewis A. Bullock, Jr.	2195

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *30 March 2007*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *19,20 and 22-38* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *19,20 and 22-38* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

Claim Rejections - 35 USC § 103

1. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

2. Claims 19, 20 and 22-38 are rejected under 35 U.S.C. 103(a) as being unpatentable over "Object Oriented Design for a Distributed Priority Queue" by Pen-Nan LEE et al. in view of "Shortest Path Algorithms" by GALLO et al.

As to claim 19, LEE teaches a method for adding a new entity (via the insert operation) having a rank (priority) within a plurality of ranks (priorities) to a plurality of entities (queues) as represented in a data structure for efficiently ordering the entitites, the entities also having respective ranks within the plurality of N ranks, the method comprising: of a plurality of table entries of a table over which the plurality of ranks are distributed such that a table entry has a corresponding range of ranks, determining the table entry having the corresponding range of ranks in which the rank of the new entity lies (see pg. 194, "The operation INSERT takes an element and a priority level for that element and determines the constituent object to which the element should be forwarded by consulting the range table. Once the destination constituent object is known, the element is forwarded to that object by invoking its INSERT operation..."; pg. 195, DPQ Object Operations, "The DPQ is divided into segments. Each segment is assigned to a different constituent in the system...Each range contains an upper and lower bound. Any element whose priority falls in between the upper and lower bound is

placed in that range...Whenever a constituent object receives an insert request message from either its local processes or other processes in the network...In the latter case, the non-empty constituent must find its proper position with respect to other non-empty constituents in the DPQ."); adjusting the table entry to point to the new entity in response to determining that the particular table entry currently points to null (pg. 195, "Initially each constituent will have the above range table and all id variables will be initialized to NIL except that the Head_ids of LDPQ2 through LPDQ5 are initialized to the id of LDPQ1....In the former case, the constituent just simple claims itself to be the head constituent of the DPQ."); adjusting the array entry to point to the new entity in response to determining that the particular array entry currently points to an entity having a rank less than the rank of the new entity (via the Hookup Operation) wherein the rank of the entity is its priority and the array is a priority queue (pg. 195-196, "A constituent issues a HOOKUP request message when its status changes from empty to non-empty. The proper position of this constituent with respect to other existing non-empty constituents in the DPQ is determined by the priority level it holds. If its priority range is the highest among the existing non-empty constituents then it will become the new head constituent. In this case, it will be placed in from of the current head constituent..."); linking the new entity into a vertically linked list linking in at least one direction a corresponding subset of the plurality of entities having an identical rank, in response to determining that the rank of the new entity is equal to the rank of any other entity within the plurality of entities (via inserting the entity on the queue) (pg. 194, and 195); and otherwise, linking the new entity into a horizontally linked list linking at least a

subset of the plurality of entities in at least a descending rank order direction, each entity in the horizontally linked list having a unique rank as compared to the ranks of the other entities in the horizontally linked list (via constituent not being empty and thereby linking the constituents in order) (pg. 195 – 196). However, LEE does not allude to the data structure being stored on a medium and the table as an array having a plurality of array entries over which ranges of the ranks are distributed. LEE does teach program code that is used to implements and manipulates the priority queue objects and a table wherein the ranges of ranks are distributed (Range_table) (pg. 194). It is well known to one of ordinary skill in the art at the time of the invention that code can be stored on a medium and therefore would be obvious for one to do so in order to save, maintain, or distribute software. It is also inherent to the teachings of LEE that since ranges of priorities are handled by respective queues that their exists less table entries then possible priorities.

GALLO teaches a priority queue structure wherein a queue is issued to refer to the individual queues / linked lists (see page 22, 7.3 Buckets; pg. 6, fig. 1). It would be obvious by one of ordinary skill in the art at the time of the invention, that the LEE's table identification of the corresponding queues and their ranges of entities corresponds to GALLO's array wherein the array entries to the various linked lists are the table entries to the various ranged LPQ's. Therefore, it would be obvious to one of ordinary skill in the art to combine the teachings of LEE with the teachings of GALLO in order to manage access to data structures thereby improving searching performance (Lee, Introduction; GALLO, pg. 3-4, abstract and introduction).

LEE and GALLO when combined teach the array is a priority queue and that the rank of an entity is its priority. However, neither LEE nor GALLO detail that the entities are threads. Official Notice is taken in that it is well known in the art that threads are entities that are queued for serviced according to some priority structure and therefore would be obvious to one of ordinary skill in the art that threads are queued using the data structure of LEE and GALLO (see for instance "Scheduling Support for Concurrency and Parallelism in the Mach Operating System" by Black).

As to claim 23, LEE teaches a data structure comprising: a plurality of entities having respective ranks within a plurality of N ranks (elements of the array / linked list); a horizontally linked list (linked list of constituents that manage a linked list of their own wherein the constituents are arranged in the list order based on the priority range) linking at least a subset of the plurality of entities, wherein each of the entities having a unique rank relative to the ranks of other entities in the horizontally linked list in at least a rank order (pg. 194, Class DPQ: Distributed Priority Queue Object, "...Each constituent object will maintain its part of the distributed state including: a range table containing the subset of ranges that each constituent object composing the distributed object is responsible for managing...a Next_id which indicates which constituent object contains the next lowest non-empty subrange of priority levels, an LPQ object for maintaining the elements which belong to the range of priorities which the constituent object is responsible for managing."; pg. 195, "Specifically, constituent objects will (among other things) need to do the following actions privately: Each constituent will

have to pass the elements which it is asked to insert into the distributed priority queue along to the constituent object which is responsible for maintaining the priority level for that element....Each of the constituents will need to communicate with each other in order to maintaining the proper ordering of elements by priority level."; see also pg. 195, DPQ Object Operations); and a table having a plurality of fewer than N table entries associated with respective ranges of the N ranks, at least one table entry each indicating an entity of the plurality of entities having a greatest rank within the corresponding range of ranks for the table entry wherein the array is a priority queue (via the table indicating the constituent that is responsible for handling a particular priority range and that constituent indicates the elements of the range) (see page 195). However, LEE does not allude to the data structure being stored on a medium and the table as an array having a plurality of array entries over which ranges of the ranks are distributed. LEE does teach program code that is used to implements and manipulates the priority queue objects and a table wherein the ranges of ranks are distributed (Range_table) (pg. 194). It is well known to one of ordinary skill in the art at the time of the invention that code can be stored on a medium and therefore would be obvious for one to do so in order to save, maintain, or distribute software. It is also inherent to the teachings of LEE that since ranges of priorities are handled by respective queues that their exists less table entries then possible priorities.

GALLO teaches a priority queue structure wherein a queue is issued to refer to the individual queues / linked lists (see page 22, 7.3 Buckets; pg. 6, fig. 1). It would be obvious by one of ordinary skill in the art at the time of the invention, that the LEE's

table identification of the corresponding queues and their ranges of entities corresponds to GALLO's array wherein the array entries to the various linked lists are the table entries to the various ranged LPQ's. Therefore, it would be obvious to one of ordinary skill in the art to combine the teachings of LEE with the teachings of GALLO in order to manage access to data structures thereby improving searching performance (Lee, Introduction; GALLO, pg. 3-4, abstract and introduction).

LEE and GALLO when combined teach the array is a priority queue and that the rank of an entity is its priority. However, neither LEE nor GALLO detail that the entities are threads. Official Notice is taken in that it is well known in the art that threads are entities that are queued for serviced according to some priority structure and therefore would be obvious to one of ordinary skill in the art that threads are queued using the data structure of LEE and GALLO (see for instance "Scheduling Support for Concurrency and Parallelism in the Mach Operating System" by Black).

As to claim 31, LEE teaches a method for removing a particular entity from a plurality of entities (via deleting the entity) of a data structure, the entities having respective ranks (priority) within a plurality of N ranks (priorities), the data structure including an array of one or more array entries (table having table entries), the method comprising: in response to determining that the particular entity is present within a vertically linked list of a subset of the plurality of entities having an identical rank, delinking the particular entity from the vertically linked list (via determining which constituent object contains the highest non-empty priority level and invoke that

constituent objects delete operation wherein that object which contains the element to be deleted performs the deletion of the element from its LPQ object by invoking the LPQ object's delete operation) (pg. 194, "The operation DELETE..."; pg. 195, 4.2 Delete Operation); in response to determining that the particular entity is present within a horizontally linked list (linked list of constituent objects) linking at least a subset of the plurality of entities in a rank order direction, delinking the particular entity from the horizontally linked list (via determining which constituent object contains the highest non-empty priority level and invoke that constituent objects delete operation wherein that object which contains the element to be deleted performs the deletion of the element from its LPQ object by invoking the LPQ object's delete operation and if the element is the last element in the head constituents LPQ, the constituent referenced in the Next_id of the head constituent should become the new head constituent) (pg. 194, "The operation DELETE..."; pg. 195, 4.2 Delete Operation)(via setting the Next_id of the head constituent as the new head); in response to determining that one of the array entries points to the particular entity, adjusting the array entry to point to one of null and another one of the plurality of entities (via communication among constituents to maintain the proper order of priority levels, therefore deleting of a constituent object will remove it from the array that points to the constituent object) (see page 195). However, LEE does not allude to the data structure being stored on a medium and the table as an array having a plurality of array entries over which ranges of the ranks are distributed. LEE does teach program code that is used to implements and manipulates the priority queue objects and a table wherein the ranges of ranks are distributed (Range_table)

(pg. 194). It is well known to one of ordinary skill in the art at the time of the invention that code can be stored on a medium and therefore would be obvious for one to do so in order to save, maintain, or distribute software. It is also inherent to the teachings of LEE that since ranges of priorities are handled by respective queues that their exists less table entries then possible priorities. LEE also teaches that when removing an entity from the queue if the queue becomes empty, then the table entry is adjusted nil (pg. 195, "For a DELETE operation, if there is at least one element...If the DELETE operation causes the DPQ to become empty, then the head constitutuent which contains the last element simply changes its Head_id to NIL.")

GALLO teaches a priority queue structure wherein a queue is issued to refer to the individual queues / linked lists (see page 22, 7.3 Buckets; pg. 6, fig. 1). It would be obvious by one of ordinary skill in the art at the time of the invention, that the LEE's table identification of the corresponding queues and their ranges of entities corresponds to GALLO's array wherein the array entries to the various linked lists are the table entries to the various ranged LPQ's. Therefore, it would be obvious to one of ordinary skill in the art to combine the teachings of LEE with the teachings of GALLO in order to manage access to data structures thereby improving searching performance (Lee, Introduction; GALLO, pg. 3-4, abstract and introduction).

LEE and GALLO when combined teach the array is a priority queue and that the rank of an entity is its priority. However, neither LEE nor GALLO detail that the entities are threads. Official Notice is taken in that it is well known in the art that threads are entities that are queued for serviced according to some priority structure and therefore

would be obvious to one of ordinary skill in the art that threads are queued using the data structure of LEE and GALLO (see for instance "Scheduling Support for Concurrency and Parallelism in the Mach Operating System" by Black).

As to claim 20, LEE teaches adjusting a head pointer to an entity having the greatest rank of a plurality of ranks of the plurality of entities to point to the new entity in response to determining that the rank of the new entity is greater than the rank of the entity of the current head pointer (via performing a HOOKUP Operation and determining the proper position of the linked list / constituents based on the priority range) (pg. 195-196).

As to claim 22, refer to claim 19 for rejection.

As to claim 24, LEE teaches the data structure comprising a vertically linked list of a subset of the plurality of entities having an identical rank (the LPQ of a constituent) (pg. 194, "Each constituent object will maintain...an LPQ object for maintaining the elements which belong to the range of priorities which the constituent object is responsible for managing.").

As to claim 25, LEE teaches the vertically linked list links the corresponding subset of the plurality of entities in a vertical direction (pg. 194, "Each constituent object will maintain...an LPQ object for maintaining the elements which belong to the range of

priorities which the constituent object is responsible for managing."). However, the cited combination does not allude to the object has having a second direction. Official Notice is taken in that it is well known in the art that double linked list are well known in the art and that the linked list queue of LEE is a well known double linked list that is capable of having two directions.

As to claim 26, LEE teaches the data structure further comprises a head pointer pointing to an entity having a greatest rank of the plurality of ranks of the plurality of entities (Head_id) (pg. 194).

As to claim 27, LEE teaches the horizontally linked list (list of constituents from head on down based on the next_id pointer) further links at least the subset of the plurality of entities in an ascending rank order direction (via the hookup operation) (pg. 195 – 196).

As to claim 28, LEE teaches the plurality of ranks are distributed over the plurality of array entries (pg. 195, fig. 1). It would be an obvious design choice that this distribution is equal.

As to claim 29, LEE teaches an array entry for a range has to elements having different ranks (via inserting an element in a linked list upon determining it falls upon a

range, such that the element is linked after an existing element) (pg. 194, 1st and 2nd columns).

As to claim 30, LEE teaches at least one table entry of the plurality of table entries each points to null, corresponding to no entity within the plurality of entities having a rank within the corresponding range of ranks for the table entry (via initializing the head_ids to nil) (pg. 195, DPQ object Operations).

As to claim 32, LEE teaches the table entry has a corresponding range of ranks, and adjusting the table entry to indicate one of null and another one of the plurality of entities comprises, in response to determining that the particular entity was present within the vertically linked list, adjusting the table entry to point to a next entity within the vertically linked list (via the head pointer remaining pointing to the head constituent which would indicate a new first element) (pg. 195, "For a DELETE operation, if there is at least one element in the DPQ, the first element of the highest non-empty priority level in the head constituent will be deleted from the constituent's LPQO.").

As to claim 33, LEE teaches adjusting the table entry to point to one of null and another one of the plurality of entries comprises, otherwise in response to determining that the particular entity was present within the horizontally linked list, and that the rank of the next entity within the horizontally linked list is within the corresponding range of ranks for the array entry, adjusting the table entry to indicate to the next entity within the

horizontally linked list (pg.195, "When a delete operation removes the last element from the head constituent's LPQ , the constituent referenced in the Next_id of the head constituent should become the new head constituent.")

As to claim 34, LEE also teaches that when removing an entity from the queue if the queue becomes empty, then the table entry is adjusted nil (pg. 195, "For a DELETE operation, if there is at least one element...If the DELETE operation causes the DPQ to become empty, then the head constitutuent which contains the last element simply changes its Head_id to NIL.")

GALLO teaches a priority queue structure wherein a queue is issued to refer to the individual queues / linked lists (see page 22, 7.3 Buckets; pg. 6, fig. 1). It would be obvious by one of ordinary skill in the art at the time of the invention, that the LEE's table identification of the corresponding queues and their ranges of entities corresponds to GALLO's array wherein the array entries to the various linked lists are the table entries to the various ranged LPQ's. Therefore, it would be obvious to one of ordinary skill in the art to combine the teachings of LEE with the teachings of GALLO in order to manage access to data structures thereby improving searching performance (Lee, Introduction; GALLO, pg. 3-4, abstract and introduction).

As to claim 35, LEE teaches in response to determining that a head pointer pointing to an entity having a greatest rank of the plurality of ranks of the plurality of entities points to a particular entity, adjusting the head pointer to point to another one of

the plurality of entities (pg.195, "When a delete operation removes the last element from the head constituent's LPQ , the constituent referenced in the Next_id of the head constituent should become the new head constituent.").

As to claim 36, LEE teaches adjusting the head pointer to point to another one of the plurality of entities comprises, in response to determining that the particular entity was present within the vertically linked list, adjusting the head pointer to point to a next entity within the vertically linked list (via the head pointer remaining pointing to the head constituent which would indicate a new first element) (pg. 195, "For a DELETE operation, if there is at least one element in the DPQ, the first element of the highest non-empty priority level in the head constituent will be deleted from the constituent's LPQO.").

As to claim 37, LEE teaches adjusting the head pointer to point to another one of the plurality of entities comprises, otherwise in response to determining that the particular entity was present within the horizontally linked list, adjusting the head pointer to point to a next entity within the horizontally linked list (pg.195, "When a delete operation removes the last element from the head constituent's LPQ , the constituent referenced in the Next_id of the head constituent should become the new head constituent.").

As to claim 38, refer to claim 10 for rejection.

Response to Arguments

3. Applicant's arguments filed March 30, 2007 have been fully considered but they are not persuasive. Applicant argues that the Office has admitted that neither Lee nor Gallo teaches or suggest any method wherein at least one entity of the plurality of entities is a thread, the rank of the entity is a priority for the thread and the array is a priority queue, but has taken Official Notice that threads as entities that are queued are well known in the art cannot be supported by a showing of adequate evidence. The examiner disagrees. The examiner would first like to make Applicant aware of a number of key issues, yet to be resolved. First, is the thread entity which has a priority, included in any of the data structures, e.g. the array, the horizontal linked list, or the vertical linked list. The claims (in particular claims 23-30) do not make this a requirement. Therefore, at the very least, the claims require that the array is a priority queue, and the elements of the array, the vertical linked list, and the horizontal linked list do not have to be the at least one thread entity because the thread entity is one of a plurality of entities and each substructure, i.e. array, linked list; only uses a subset of the entities that do not necessarily include the thread entity. Secondly, The prior art of Lee is directed to the array/linked list data structure implementation as a distributed priority queue, so the rank of the entities are the priority and the array is a priority queue. The only missing element is that the entities themselves are threads. "Scheduling Supprt for Concurrency and Parallelism in the Mach Operating System" by Black is an illustration that threads as entities in a priority queue are well known in the art. Specifically,

throughout page 37, Black teaches priority queues containing threads based on their priority. Therefore, the rejection is maintained and met as detailed in the action. Regarding all the other claims, Applicant has made the same argument and therefore, the same reasoning and logic as detailed herein applies to those claims in maintaining the rejection.

Conclusion

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Lewis A. Bullock, Jr. whose telephone number is (571) 272-3759. The examiner can normally be reached on Monday-Friday, 8:30 a.m. - 5:00 p.m..

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Meng An can be reached on (571) 272-3756. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

June 7, 2007

LEWIS A. BULLOCK, JR.
PRIMARY EXAMINER